95-702 Distributed Systems Project List Summary

This list represents a fairly close approximation of the projects that we will be working on. However, these projects are subject to change as the course proceeds.

Each of these projects act as a small exemplar of real world distributed systems.

1. JEE Web Application

The project may be demonstrated in class and serves as a jumping off point to discuss important characteristics of distributed systems including security, protocol layering and interoperability.

This project has three parts:

First, the student is introduced to GlassFish. GlassFish is an open source application server that implements the JEE 5 specification. This tool is used throughout the course. The Netbeans integrated development environment is introduced and is used to build source code and interact with GlassFish.

Second, the student builds their first distributed system. The student builds three small web applications using Java Server Pages and servlets.

Third, the student is introduced to the Android simulator. The simulator runs within the Eclipse IDE. In this first Android project, only the browsing capabilities of Android are explored.

For this project, the non-functional characteristics that need to be addressed in your paper include security, protocol layering and interoperability.

2. JEE Web Services

The project may be demonstrated in class and naturally leads to discussions concerning concurrency, state maintenance and interoperability.

This project has three parts:

First, the student is exposed to TCP/IP socket programming. An interactive and stateful distributed system implemented with TCP sockets is presented. The protocol is described with a finite state machine and written in Java. The student is asked to build a web application that does much the same thing. The interactive socket based client and server is replaced with browsers and servlets. The protocol handler itself, is simply reused in the student solution.

Second, this project introduces state management and concurrency. The stateful web application that is constructed by the student must be usable by several concurrent visitors. Each visitor will have its own protocol handler on the server.

Third, the project introduces web services. The student is asked to rebuild the project once again. The browser is removed and replaced with a web service client and the servlet is removed and replaced with a SOAP based web service. State management using cookies is replaced with a user identifier.

For this project, the non-functional characteristics that need to be addressed in your paper include concurrency, state maintenance and interoperability.

3. TCP, UDP and Distributed Objects

The project may be demonstrated in class and naturally leads to discussions concerning protocol layering, UDP, TCP, IP, interoperability, marshaling, external data representation, naming, separation of concerns and design patterns.

This project has two parts:

First, the student is exposed to both TCP and UDP sockets. Several simple programs are written that illustrate how these protocols are similar and how they differ. These simple exercises are designed to provide the students with skills needed for the second part of the project.

Second, the student is exposed to issues associated with remote method invocation. Code is provided that acts as sort of scaffolding upon which the student builds a more significant application. The scaffolding includes a client side proxy and server side skeleton. Using that code as a model to work from, the students write a simple registry and an object server and client. The program uses sockets but is careful to illustrate separation of concerns and the use of the proxy design pattern. Ideas from the Colouris text, such as remote object references and request reply messages, must be implemented by the student.

For this project, the non-functional characteristics that need to be addressed in your paper include protocol layering, marshaling, interoperability, external data representation, naming, separation of concerns and design patterns.

4. Synchronous and Asynchronous Java RMI

The project may be demonstrated in class and naturally leads to discussions concerning synchronous and asynchronous calls, event handling, remote interfaces, remote interface compilation, interface definition languages and the publish/subscribe design pattern.

This project has three parts:

First, code is presented that acts as a scaffolding for the remaining parts. A complete Java RMI solution is presented and the student is asked to build and run the system using the directions provided. The student is taught exactly how to use the rmi regsitry and the rmic tool.

Second, using Java RMI, the student builds a distributed but synchronous chat server. Several clients must be shown to be running and chatting. The solution is modeled after the distributed whiteboard application that is found in the Coulouris text.

Third, the student is required to add asynchronicity to the chat server. This amounts to providing a call back handler that runs on each client. This improvement makes for a livelier user interaction and illustrates the importance of event handling.

For this project, the non-functional characteristics that need to be addressed in your paper include synchronous and asynchronous calls, event handling, the importance of remote interfaces, remote interface compilation, interface definition languages and the publish/subscribe design pattern.

5. Web Services and JDBC Transactions

The project may be demonstrated in class and naturally leads to discussions concerning WSDL, interoperability, locks, transaction handling, entity beans and distributed transactions.

This project has three parts:

First, the student is provided with detailed instructions on building a relational database using Netbeans and Sun's Java DB. The student is also provided with code that acts as a wrapper around a single row of the database. The code makes use of JDBC to interact with the database. The student is required to run the code and see the impact on these data.

Second, the student is provided with a new database schema and is required to create a small database with several tables and several constraints on database fields.

Third, the student is asked to write a SOAP based web service that may be used by two web service clients to make updates to the data. One client is a JSP page running on Glassfish and the other client is a standalone console based application. If a constraint violation occurs, the web service is written to rollback the local transaction. This is done with the rollback and commit features of JDBC connections.

For this project, the non-functional characteristics that need to be addressed in your paper include interface definitions using WSDL, interoperability, locks, transaction handling, entity beans and distributed transactions

6 Messaging

The project may be demonstrated in class and naturally leads to discussions concerning messaging, asynchronous services, interoperability, as well as reliability and message persistence.

This project has three parts:

Part one consists of a short tutorial on writing and running a JMS application using Netbeans and Glassfish.

Part two requires that student write a message driven bean that takes a java text message off a JMS Queue.

Part three requires that the student write message handlers for JMS Topics.

For this project, the non-functional characteristics that need to be addressed in your paper include messaging, asynchronous services, interoperability, reliability, lose coupling and the publish subscribe paradigm.